ENCRYPTION AND AUTHENTICATION SYSTEMS AND METHODS

Inventor: Edwin A. Suominen

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims benefit of U.S. provisional application 60/273,862, filed March 5, 2001 and having the same title and inventor as the present application. That provisional application, including all materials incorporated by reference therein and its appendices, is incorporated herein by reference in its entirety.

BRIEF DESCRIPTION OF THE DRAWINGS

This application includes numerous drawings where necessary for the understanding of the subject matter sought to be patented, though they are not presently set apart in a separate group of drawing sheets of sequentially numbered figures. Instead, drawings pertinent to textual disclosure of particular aspects of the inventions are interspersed with that text throughout the appendices.

SPECIFICATION

Introduction

The bulk of this specification resides within numerous attached appendices, which are incorporated by reference herein and form an integral part of the specification. Each appendix is labeled with a letter, starting with "A" and ending with "AL." (Appendixes beyond "Z" have two letters, starting with "AA.") The appendices are referenced below by the word "Appendix" followed by the appendix letter and, optionally, one or more page numbers within the referenced appendix. The word "Appendix" may be omitted in places where it is understood by the context of the reference.

The description below is largely directed to preferred exemplary embodiments of the applicant's various inventions relating to encryption and authentication systems and methods. This section is accompanied by a section below entitled "Exemplary Elements of the Inventions," which disclose exemplary elements intended to support appropriately

broad future claim language, but which are not claims. (Paragraphs introducing each overview of each respective aspect are numbered for convenience only.)

This application includes a single claim, which is found just before the appendices.

Specificity of language in this informal specification (particularly the appendices) does not imply any limitation on the scope of any of the applicant's inventions. The broadening terminology that often appears in formal patent applications such as "preferably," "in a variation," "in one embodiment" etc. may in places be absent. However, portions of the application introducing structural and method elements of the various inventions should be understood as including such terminology.

Thus no one embodiment disclosed herein is essential to the practice of another unless indicated as such. Indeed, the applicant contemplates that his inventions, as supported by the disclosure of this informal specification and appendixed drawings, include all systems and methods that can be practiced from all suitable combinations of the various aspects disclosed, and all suitable combinations of the exemplary elements listed. (As but one example, the applicant views the implementation of a secure delay with a pseudogroup operation as a particularly advantageous combination according to one aspect of his inventions. See Appendices K and L for specific disclosure of this particular combination.) Such combinations have particular advantages, include advantages not specifically recited herein.

The applicant contemplates that alterations and permutations of the preferred embodiments and methods and will become apparent to those skilled in the art upon a reading of the specification and a study of the appendixed drawings. Accordingly, none of the disclosure of the preferred embodiments and methods below defines or constrains the inventions. Rather, the issued claims variously define the inventions.

The "Covenant Trust" Authentication System

1.Authorization and Certification Instrument ("ACI") that, with a holographic (ink) signature, legally binds the signer to digital signatures uniquely corresponding to (and

thus created with) a positively identified public key. See example shown in Appendix A-1

2. Method of certifying a subscriber's public key, comprising: (a) having the subscriber execute an ACI; (b) confirming that the ACI has been signed and notarized in ink; and (c) publishing the confirmation for persons relying on the public key.

See flow diagram of an exemplary process in Appendix B and a description of the method in Appendix C. Appendix C is adapted from a confidential document describing benefits of the inventive method (as a technology rather than as a product or service) generally and, in conjecture only, a specific example of the method. The specificity of the example is not intended to any way limit the scope of the invention or imply that any product or service described in Appendix C has been offered for sale as of this application's filing date.

3. Method of authenticating a subscriber, comprising: having the subscriber agree to terms of the ACI (directly or by reference to a ACI displayed on a secure web page) in a recorded telephone conversation. See Appendix D for an example. See Appendix E for a disclosure of a technique for authenticating the recording.

One advantage of this method over the use of a holographically-signed paper ACI is that a person relying on the subscriber's digital signature is likely to know the sound of the subscriber's voice, and is thus likely to trust a recorded verbal agreement. SelfCertify.com can sign a digital file containing the recording (e.g., in compressed WAV format) as it can a PDF/TIFF file of a paper ACI. A recorded "verbal ACI" can have security features somewhat analogous to those of a paper document. For example, a synthesized voice can recite digits of the fingerprint of a public key certified by the ACI repeatedly throughout the entire recording, preferably with subdued volume.

4. Method of entering into an agreement (e.g., an NDA) concerning an electronic record (e.g., a confidential disclosure) comprising: (a) positively identifying the information (e.g., by sending it encrypted, by sending a hash of it, etc.); and (b) including the positive identification in the agreement. A particular method includes

sending the record in encrypted form and, upon entering into an NDA with the recipient, sending the decryption key for the record.

5. Positive identification of an electronic record with an "integrated" combination of: (a) a code "uniquely corresponding" to that electronic record (e.g., a SHA-1 cryptographic hash code); and (b) a holographic signature and (or facsimile thereof). The combination is said to be integrated when it would be difficult for a forger to separate the elements of the combination. Another way of describing an integrated combination of (a) and (b) is having a contextual thread between (a) and (b). (See the discussion of the exemplary element "background digits" below.) A paper document or facsimile copy thereof containing the combination can include the following advantageous aspects:

- The digits of the code can be printed in background digits of the document, including behind fields for handwriting. See Appendix A for an example of an ACI with background digits of a PGP "fingerprint."
- The document can include a facsimile copy of a photographic identification of the signer, which can be referenced in language of the document (e.g., a Notary's statement). For example, an ACI can include a photocopy of a driver's license.

The ACI (Appendix A) is a specific example in which the electronic record is a public signing key. Another example, in which an electronic copy of a publicly accessible paper file is positively identified by a third party who has inspected the file, is shown in Appendix F.

6. Signed statement authenticating an electronic representation of a document or tangible item. See examples shown in Appendix F (certified electronic copy), Appendix G (certified reproduction), and Appendix H (certified contents of Express Mail envelope).

See, generally:

- Appendix A: An exemplary ACI in which various features listed above are implemented.

- Appendix C: A description of the "Covenant Trust" authentication system, describing embodiments of the ACI and hardware/software implementation of an exemplary system using it.

THE PSEUDOGROUP OPERATION AND VARIOUS USES OF IT IN CRYPTOGRAPHIC PRIMITIVES

7. z = x*y mod p, where p = 2^N+k (prime, preferably lowest prime greater than 2^N) and one of the following actions are taken in the event (rare when k<<2^N), when the output z > 2^N:

- The output z is mapped to a "hole" in the output set. A hole is a value that will not ever occur for a particular key, with any input from the set {1,2,...2^N}. This option is particularly advantageous in that it is fully reversible - each possible input maps to a unique output, and vice versa; or
- The output z is simply allowed to overflow, or is assigned a value according to a predetermined rule (e.g., z = x, z = y, etc.). When decryption needs to be done and k << 2^N, the error (from uncertainty as to which input caused the output) can be made less likely than other sources of bit error and handled with standard error-correcting schemes.

8. z = x*y mod p, where p = 2^N-k (prime, preferably greatest prime less than 2^N) and inputs x >= p are passed through without multiplication by y. Keys y and inverses y^-1 can be limited to set {1,2,...2^N}. Alternatively, overflowing key or inverse key values can simply be adjusted (e.g., by zeroing an MSB or subtracting a constant like c = p-2^N) if y >= p.

Note that the probability of any given input value being in the pass-through range between p=2^N-k and 2^N is very small if k << 2^N, a condition that can usually be achieved. Also note that a single pass-through of an input value will not significantly degrade the security of a multiple-round cipher or secure delay primitive. See Appendix I, a MATHCAD analysis illustrating how vanishingly small the probability is of encountering repeated pass-through values, with k << 2^N.

9. Alternative to the IDEA cipher (see the '703 patent referenced below), preferably with 32-bit sub-blocks (instead of 16-bit sub-blocks) but, in variations, scalable to any desired block length. With preferred 32-bit sub-blocks, total block length is 128 bits (instead of IDEA's 64 bits), and modulus is preferably p = 2^32+15 (see Item 7 above) or p = 2^32-5 (see Item 8 above). See, in Appendix J, a diagram of a straightforward modification (Appendix J-5) to the IDEA cipher with a 128-bit block length and a 32-bit version of the inventive pseudogroup modulo product operation; and annotations to IDEA-related excerpts (Appendix J-4,5,16) from cryptography texts.

10. Secure delay primitive using pseudogroup operation mixed with at least one group operation. The following are various exemplary embodiments with delay blocks comprising:

- Appendix K-1: (1) pseudogroups and (2) a conventional group operation (e.g., x+y mod 2^N or bitwise modulo addition, XOR), cross-connected in a manner structurally similar to the cross-connection of blocks in the IDEA cipher.
- Appendix K-2: Sequentially but in no particular sequence: (1) a pseudogroup operation, (2) x+y mod 2^N, and (3) bitwise modulo addition (XOR).
- Appendix K-3: an embodiment including: (1) a pair of pseudogroup operations; (2) a pair of S-boxes coupled to the respective pseudogroup operations; and (3) a diffusion layer to mix up the paired outputs into an overall output value for the delay block.
- Appendix L: (1) a multiple-precision pseudogroup product of four inputs; and (2) respective S-boxes. In this embodiment, a conventional x*y mod 2^16+1 operation (as disclosed in U.S. Patent 5,214,703 to Massey et al., incorporated herein by reference) can be used with an overall block size of 64 bits.

See, regarding the p = 2^N +/- k pseudogroup operation:

- Appendix M: A listing of primes just above and below powers of two. Two sets of primes (not shown in the listing) are especially useful for implementing a secure delay of 160-bit SHA-1 hashes outputs. Those are 2^80-65, which lends itself to

efficient modular reduction because 65 has only two non-zero bits (see Appendix AL), 2^80+13, 2^160-47, and 2^160+7, which is amazingly close to 2^160. (xy mod 2^160+7 is vanishingly unlikely to have overflows. One variation, useful in communications applications, relies on error correction encoding to correct errors caused by rare overflows in such "overflow unlikely" cases.

- Appendix N: Spreadsheet printouts showing the output values of pseudogroup operations over moduli 2^3+3, 2^4+3, and 2^5+5, and the locations of holes for the keys in the respective sets for those operations. (The deterministic positioning of holes is clearly visible.)
- Appendix O: A mathematical derivation of exemplary equations for encryption and decryption.
- Appendix P: Illustrates results of a hole-plotting Octave script. P-1,2,3 illustrate output values (along the row axis perpendicular to the ones and zeros parallel to the handwritten writing) that are holes. The holes are values that do not occur within the set S:{1,2,...64}, given a particular key value (columns labeled 1-64) within set S, as the result of a modulo product with modulus > 64. In P-1, the modulus is 71. In P-2, the modulus is 67. In P-3, the modulus is 73.
- Appendix J: Disclosure printout that includes, inter alia, illustrations (Appendix J-6,7) of p = 2^N-k pseudogroup output values, given all inputs ("y" axis -- increasing input values in lower rows) and keys ("x" axis - increasing key values in rightmost columns).

See the following regarding an actual "software prototype" implementation of the p = 2^N+k pseudogroup operation:

- Appendix Q: Octave functions used in the software.
- Appendix S: Octave diary showing console output with results of the successful test of a hole location function according to various aspect, of the inventions. Appendix R-2 lists Octave code for a brute-force hole-finding function according to a less preferred embodiment. Appendix R-3 lists Octave code for a particularly advantageous hole-finding function according to various aspects of the inventions.

Also included in Appendix R-3 is commented-out code with a hole-finding function that is very straightforward but occasionally produces an incorrect result. Appendix R-4 through R-11 show the console output of the test, with only the first and last few results of the test output shown and some portions redacted. Note that there was no difference found between hole locations identified with the brute-force function and the particular advantageous function of Appendix R-3.

- Appendix S: An alternative, less preferred hole location function.
- Appendix T: A script ("TEST3.M") that executed a test of encryption and decryption using the p = 2^10+7 pseudogroup.
- Appendix U: Octave diary showing console output with results of the successful test performed with TEST3.M (Appendix T). (Only the first and last few results of the test output are shown, and some portions of the console output are redacted.) Note, for each possible key, the following successful results, indicated by + signs in columns (x) of Appendix U-5 through U-12: (2) all outputs fell within the input set {1,2,...1024}; (2) all elements in the output set {1,2,...1024} were unique; and (3) all elements in the input set {1,2,...1024} were able to be transformed (see Appendix Q-3) into outputs in {1,2,...1024} with the pseudogroup and the given key, and then transformed (see Appendix Q-4,5) back into the identical input value with the pseudogroup and an inverse key. (The tested condition for the fourth column was not met, but applicant does not consider that condition significant in the practical use of the pseudogroup. See Appendices V and W.)
- Appendix V: Octave diary showing console output with results of a specialized test performed with TEST3B.M. (Only the first and last few results of the test output are shown, and some portions of the console output including date information are redacted.) This specialized test was designed to quantify a phenomenon observed in the results of TEST3.M in which a given input, encrypted with all keys from the set of possible keys, produces duplicated outputs for two or more keys in the set. TEST3B.M empirically confirmed the inventor's hypothesis that the number of duplicated outputs is proportional to k in the modulus definition p = 2^N+k.

- Appendix W: A plot showing results of a modified version of TEST3B.M with p = 2^11+5.

USER-FRIENDLY REPRESENTATIONS OF HIGH-ENTROPY DATA WORDS

11. A "consonant-vowel" ("CV") passphrase of repeated pairs of consonants and vowels. The consonants are preferably selected to be phonetically and visually distinct. The set of consonants {b,d,g,h,k,l,m,n,p,r,s,t,z} has particular advantages. The use of "pseudowords" created from four adjacent CV pairs is particularly advantageous because the "pseudowords" are pronounceable and have a linguistic look and sound. See the following files:

- Appendix X: "Pronounceable Passphrase Worksheet" for selecting a "CV" passphrase. Note that the "worksheet" can alternatively be a computer display, package panel, etc.
- Appendix Y: Lists entropies of this type of passphrase along with others.
- Appendix Z: Discloses various aspects of secure passphrase entry according to the inventions, including graphical entry of a "CV" passphrase.

12. Thirteen-bit segments represented as a decimal values in a set of 8192 values. Preferably, the range is continuous from 1000 through 9191. By using a minimum value of 1000, confusion about leading zeros is avoided. This representation is currently considered preferable for "sub hash" outputs of a securely delayed hash (see Appendix AA).

13. Digits from a set of the letters A-N and P-Z, and all the decimal digits 1-9 and zero. The arrangement is particularly advantageous because: (1) the digits can be displayed in a 7x5 matrix, and (2) confusion between the letter "O" and zero is avoided. A particularly advantageous passphrase entry system using this "Alphanumeric-Less-O" passphrase type, along with an analysis of entropy using the preferred non-repeating of digits in that system, is found in Appendix AB.

See, generally:

- Appendix Y: A spreadsheet printout showing various types of passphrases (including conventional ones) with their entropies.

CRYPTOGRAPHIC SYSTEM ENHANCEMENTS - PERFORMANCE

14. Secure generation of passphrases (or other secure random values) by having a user randomly select digits in a field of displayed digits. Advantageously, the randomness of digits selection is entirely independent of any computer. Thus, security concerns are avoided about non-random number generation and third-party snooping or control over digit values. In a particularly simple and advantageous embodiment, the field of digits is printed out on a piece of paper (see Appendix X for example) and the user selects digits of the passphrase by tossing an object (preferably a partially an unbent paper clip) onto the piece of paper.

15. Secure delay processing of hash codes and digital signatures. (Delaying passphrase input for enhanced security, per se, is known .) See Appendix AA for details on secure hash delay.

16. Secure delay: step-by-step delay processing. (See Appendix Z) By performing delay processing in steps and incrementally accepting user input (or displaying user output), the secure delay can provide the security benefit of increased intractability to an attacker without creating any noticeable inconvenience for the legitimate user. A secure-delayed passphrase entry system can process each keystroke (or graphical digit selection) input to produce succeeding delay output values, each of which is used as the initial value for the subsequent secure delay processing of the next input. A secure-delayed hash output system can display sub-hashes of an iteratively updated delay output value so that the user can visually compare digit clusters (e.g., "1234", "X7J", "zasadabi", etc.) while the secure delay is running.

17. Secure delay: optimized efficiency for legitimate user. By keeping an attacker from being able to mount a more efficient implementation of the secure delay than the implementation available to a legitimate user with his or her desktop PC, a system

according to various aspects of the inventions enhances security with minimal delay to the legitimate user. According to this invention, efficiency is optimized not by stringing together a bunch of "junk code," which offers many opportunities for attackers to find shortcuts with optimized ASIC emulations, but by repeated application of selected mathematical operations (e.g., 80 or 160-bit pseudogroup products interspersed with modulo sum and/or bitwise XOR operations, see Appendix J). The operations are preferably chosen to be simple, easily published, and efficient, with a history of being optimized through years of published research in the cryptographic community and can be expected to be efficiently performed by the user's CPU (e.g., Pentium III, PowerPC).

See, regarding secure delay:

- Appendix Z: block diagrams (Z-7, Z-8), description (Z-1), and analysis of crack intractability (Z-9, Z-10).
- Appendix AC: block diagrams of various embodiments.

18. Hardware delay processor. See Appendix AJ-3 and Appendix AB for disclosure of various embodiments: one including a keypad or touchpad for passphrase entry. Various embodiments of Appendix AJ-3 are listed in the following table:

Label	Notes
A	In-line with keyboard. Looks for password hotkey and calculates big passphrase, updating with each entered keystroke, until "Enter" key. Then "types" big passphrase. See Appendix AJ-1.
B	Interfaces with PC through USB driver. Can implement secure delay for passphrases or hashes. Driver can emulate keyboard (for delayed output) to interface with all software. Unobtrusive hardware can also serve as a USB extension cord or hub.
C	Standalone device, with keypad for entering passphrase digits. Preferably uses standard USB keyboard driver. See Appendix AB-2.
D	Delay can be implemented in PC, with no hardware acceleration.

Label	Notes
E	Delay can be implemented in central server with very fast processing, although there would be security concerns for passphrase delays. (Might be good for hash delays, though.)
X	The software driving the hardware delay processor can transmit, as an encoded audio burst (~1/2 sec.) via the PC speakers, a symmetric key to the hardware delay processor to encode the delayed (expanded) passphrase. The software then decodes the encoded passphrase using the symmetric key, which only the software knows. Thus, the passphrase cannot be "sniffed" from USB packets or keystrokes.

CRYPTOGRAPHIC SYSTEM ENHANCEMENTS - USABILITY

19. Signing an electronic record (e.g., MS WORD document, AUTOCAD drawing, etc.) by "printing" that record to a special "virtual signature printer." See Appendix AD. The "virtual signature printer" provides the user with an intuitive, simple way of authenticating an electronic record.

20. Embedded Signature: graphical indicia of signature placed within a graphical (image-based) representation of an electronic record. See Appendix AE.

21. Embedded Signature: signature data placed within unused or backward-compatible field of an electronic record. See Appendix AF.

22. Instead of requiring all keys to be in a single "key ring" file, links to "indirect" keys in various locations can be used. See Appendix AG.

23. Graphical (mouse) input of "CVCVCVCV" passphrase digits. See Appendix Z.

24. Graphical (mouse) input of "Alphanumeric-Except-O" passphrase digits in a 7x5 grid, with already-selected digits indicated. See Appendix AB.

25. Preserving formatting (e.g., font size, boldface, underlining, etc.) of clear-signed text. See Appendix AH.

26. Unobtrusive digital signature around clear-signed text. See Appendix AH. A digital signature can be made unobtrusive in a number of different ways including:

- Using a small, distinct font (e.g., 8 point Courier) for the Base-64 encoded signature characters.
- Omitting the "header" that tells conventional digital signature software where to start looking for the beginning of clear-signed text. In the inventive alternative to such a header, the signature block (at the end of the clear-signed text) includes the number of characters in the clear-signed text, which allows the digital signature software to "count backwards" from the end of the clear-signed text until it reaches the beginning of that text.
- Using a white font color to hide the signature characters entirely.

CRYPTOGRAPHIC RECORD DESTRUCTION

27. Periodic purging of documents and/or e-mail messages by mutually agreed and/or scheduled destruction of shared keys. See Appendix AI and AJ-1.

28. A hardware electronic record "lock" with a disposable key token, such as a simple paper card with a bar code printed on it. See Appendix AJ. The key token can be thrown away when the "locked" (i.e., encrypted) electronic record is to be purged by making decryption of it practically impossible.

RANDOM BIT GENERATOR

29. A random bit generator that derives its entropy from the skew between a PC's CPU clock (due to PLL phase noise) and its crystal-controlled real-time clock. See Appendix AK.

SUPPLEMENTAL DESCRIPTION OF PARTICULAR EMBODIMENTS

THE PSEUDOGROUP OPERATION

(See items 7-10 above and appendices referenced therein.) The multiplicative group xy modulo p (where p is prime) has many applications, particularly in the field of

cryptography. Any unique combination of two multiplicands (i.e., inputs) from a set S:{1,2,...M} multiplied modulo p, where p = M+1, produces a unique result (i.e., output) in the set S. When p is large, the particular combination of inputs used cannot be easily determined based on a given output. The IDEA cipher exploits this property by using a multiplicative group modulo p = 2^16+1 (which is prime) along with two other group operations to encrypt binary data in the set {1,2,...2^16}.

Very few known moduli have the desirable property of being exactly one greater than a power of two. Appendix J-16 illustrates the undesirable lack of scalability in the IDEA cipher that results from this fact.

A pseudogroup operation according to various aspects of the present inventions dispenses with the need for the modulus to be fixed with respect to the order of the input and output set. Advantageously, the modulus can be chosen as whatever prime number is closest to the set order -- above it or below it. As may be better understood with reference to Appendix J-5, for example, advantageous modifications of the IDEA cipher according to the inventions employ moduli 15 greater than 2^32, and 5 less than 2^32, respectively, to encrypt a block of binary data in the set {1,2,...2^32}. Exemplary prime numbers for various bit lengths are listed in Appendix J-2,3 and, with some duplication, in Appendix M-1,2.

As discussed above, a pseudogroup with p > M preferably does exception handling by mapping overflowing outputs to holes. Appendix P illustrates results of a hole-plotting Octave script. Appendix P-1,2,3 illustrates output values (along the row axis perpendicular to the ones and zeros parallel to the handwritten writing) that are holes. Here, the holes are values that do not occur within the set S:{1,2,...64} as the result of a modulo product with modulus > 64, given a particular key value (columns labeled 1-64) within set S. In Appendix P-1, the modulus is 71. In Appendix P-2, the modulus is 67. In Appendix P-3, the modulus is 73.

The lines on the plots of each page illustrate the deterministic placement of holes given different key values within set S. In the example of Appendix P-1, a key value of 9 will have holes (i.e., non-occurring output values) of 62 (k=1), 53 (k=2), 44 (k=3), 35

(k=4), 26 (k=5), and 17 (k=6). Modulo 71 products that exceed 64 (one for each hole) can be advantageously mapped to a particular hole, depending on the amount of overflow in excess of 64. For example, an output of 65 (k=1) with a key value of 9 can be mapped to the value 62. An output of 66 (k=2) can be mapped to the value 53. Thus, a pseudogroup (p=71) output of 62, given a known key value of 9, can be deterministically converted back to an input value of 23, whose modulo product (unmapped) is 65.

A pseudogroup with modulus p < M preferably performs exception handling by permitting input values greater than p-1 to simply pass through unaltered. As illustrated in Appendix I-1, this exception handling can be made very rare with suitable selection of M and p. Consider the example of Appendix I-2, where M = 2^50 and p = 2^50-7. (This may or may not be a suitable prime modulus, but it serves the purpose of illustration in this example). Here, the likelihood of processing an input greater than p, given uniform random distribution of input values in {1,2,...2^50}, appears to be only 0.004 after nearly 10^12 iterations! With this low level of error, the straightforward alternative of simply allowing the erroneous output to occur and permitting conventional error-correction coding to compensate for it becomes attractive.

When performing a pseudogroup operation with p < M, key values should be less than p. Examples of such operations, with unrealistically small but illustrative values of p, are shown in Appendix J-6. The tables on that appendix page show outputs given legal key values of 1-12 and 1-6 and legal input values of 1-16 and 1-8. Appendix J-7 provides an example of a p < M pseudogroup operation with a more realistic value of p=2^24-3. (Of necessity, most elements of this example's output table are omitted.)

The "linear exception region" of the p < M pseudogroup operations of Appendix J 6,7 may be better understood with reference to MATHCAD plots in Appendix J-8 through J-15.

Throughout this application, binary sets are typically referred to as beginning with one and ending with a power of two, e.g., {1,2,...2^32} or a variable M that may be a power of two, e.g., {1,2,...M}. An actual 32-bit binary word has values ranging from

zero through 2^32-1, so suitable transformation operations may be employed, such as a simple +1 addition to the binary data (resulting in a word with one 33-bit value, 2^32 prior to each pseudogroup operation. Alternatively, an entire cryptographic operation (e.g., multiple iterations of pseudogroup and/or group operations) can be carried out with one extra bit of precision to facilitate the {1,2,...2^N} format. The output of the operation can then be converted back into the data word with one less bit by the simple subtraction operation -1 and suitable accounting for it at other portions of processing.

The pseudogroup operation, like the conventional multiplicative group of modulo p, relies on a modulo product of two numbers. This product can be implemented with a regular product followed by a modular reduction (i.e., mod p) operation, or by modular multiplication. Any suitable modular reduction or multiplication technique can be employed. Because p = M +/- k, and k << M, Algorithm 14.47 (see Appendix AL-1,2) from the Handbook of Applied Cryptography (incorporated herein by reference in its entirety) can be advantagously employed.

As discussed above, Appendix V includes an Octave diary showing console output with the first and last few results of a specialized test performed with TEST3B.M. This specialized test was designed to quantify a phenomenon observed in the results of TEST3.M in which a given input, encrypted with all keys from the set of possible keys, produces duplicated outputs for two or more keys in the set. TEST3B.M empirically confirmed the inventor's hypothesis that the number of duplicated outputs is proportional to k in the modulus definition p = 2^N+k.

TEST3B.M computes the maximum number of skipped and repeated output values for a given input and all possible keys within the set {1,2,...2^N}. The conclusion of TEST3B.M's output, at the end of Appendix V, shows that the greatest total number of skipped and repeated output values for any given input is 12, which is 2*(k-1).

Appendix W is a plot showing results of a modified version of TEST3B.M with p = 2^11+5. The plot shows skipped and repeated output value totals for each possible input in the truncated set {1,2,...1678}, each input being tested with all possible keys in

the full set {1,2,...2048}. Note that the maximum number of skips plus repeats is 8, which is also 2*(k-1) since k=5.

Having skipped output values for a given input, given all possible keys, effectively reduces the number of possible outputs of a secure delay using a pseudogroup, but by a minuscule amount (with k=5, 8 or less eliminated possibilities from a total of 2048). In a plaintext-aware attack, an attacker could cross-reference skipped output values against a known output value (i.e., a segment of cryptotext) to determine which keys not to test for against a known input value (i.e., the corresponding segment of plaintext). Again, however, the reduction (<=8) in the number of candidate keys is minuscule.

A question that may be asked is whether multiple inputs in a plaintext-aware attack could be correlated against respective outputs to further reduce the number of candidate keys. To answer the question, consider the worst possibility, in which every input in the attacker's collection of plaintext/cryptotext collection for the key under attack knocks out a unique group of 2*(k-1) possible keys. In the example of Appendix W where k=5, the number of keys to be searched in this worst-case scenario (at least the applicant cannot envision a worse one) is reduced by 8*N, where N is the number of plaintext/cryptotext number pairs in the attacker's possession. Assuming a key length (for an individual pseudogroup operation) of 2^32, the attacker must have 2^20 pairs (i.e., a 1 MB file) to reduce the number of candidate keys by about 840,000, or only 0.2% of the total. One lesson to be learned from this mental exercise, however, is that the closest possible modulus of value the 2^N boundary (on the desired side of the boundary) should preferably be employed.

In a modification of the IDEA cipher according to various aspects of the inventions, where multiple rounds of the pseudogroup are employed with isolation by conventional group operations, any vulnerability introduced by output value skipping over the key space is further reduced in significance.

The next question that may be asked is whether having repeated output values for a given input, given all possible keys, creates security concerns. The effect of repeated output values is to create multiple sets of candidate keys for a given input/output

combination. This effect may well strengthen security by increasing the amount of searching required for plaintext-aware attack, perhaps effectively offsetting any infinitesimal degradation caused by output value skipping.

EXEMPLARY ELEMENTS OF THE INVENTIONS

The following terms are said to be elements of the inventions because, in various combinations (and in some cases standing alone), they recite subject matter that the applicant views as particularly pointing out various aspects of his inventions. The terms are to be broadly understood as including any mathematical construct, structure, method, system, etc., as the case may be, suitable for carrying out the function(s) discussed.

- "ACI," "Authorization and Certification Instrument." A legal instrument executed by a conventionally accepted technique (e.g., handwritten ink, a witnessed audio or video recording, etc.) that does not rely on digital signature technology. The instrument includes a covenant not to repudiate a particularly identified digital signature key of the signer, making electronic records authenticated with that digital signature key as valid (assuming the court accepts the certification) as the conventional signature on the ACI. See Appendix A and C-3.
- "Background Digits." Digits that can be read in the background of a document "behind" other alphanumeric indicia. In a particularly advantageous arrangement, background digits are oriented perpendicular to the main orientation of the document. (See the large background digits in Appendix A.) In another particularly advantageous arrangement, small background digits fill a substantial portion of one or more document signature, stamp, and/or annotation fields. (See, in Appendix A, the small background digits in the signature fields where "VOID" is handwritten in place of actual signatures.) Background digits according to various aspects of the inventions advantageously maintain a contextual thread between (1) inked indicia (e.g., one or more holographic signatures, a notary stamp, etc.) in a document and (2) printed indicia elsewhere in the document, making it difficult to "lift" the image

of such inked indicia and transfer it to a forged document. The background digits are preferably in an outline font to maintain readability of the main document indicia. The spacing, font size, and font type of the digits is preferably varied in a way unpredictable to a forger (e.g., pseudorandomly) to make duplication of a "erasing negative" of the background digits difficult. (An erasing negative could conceivably be used to eliminate background digits and permit "lifting" of inked indicia that would otherwise have the digits in its background field.)

- "Fingerprint." A number, typically fairly long (e.g., 160 bits), uniquely corresponding to an electronic record (e.g., a signing key, encryption key, dual-function key, computer file, etc.) Often, a fingerprint is simply a hash (e.g., SHA-1) of the electronic record. According to aspects of the present inventions, a fingerprint can be encoded into a user-memorable form, such as that used in the pronounceable passphrase aspects of the inventions, or a familiar-type number resembling a telephone number, ZIP code, etc. See Appendix AA and Item 12 above.
- "Graphical input." Selection of digits based on a graphical display, using a conventional pointing device such as a mouse.
- "Hole." A number within a finite set S:{1,2,...M} that will not appear as the output of any product modulo p if p>M and the inputs (i.e., multiplicands) to the product lie within set S. According to one aspect of one invention (see item 7 above), outputs from such a modulo product that are greater than M and thus lie outside the valid set of numbers S are mapped deterministically to a particular hole, selected according to how much the output exceeds M. Such mapping will not cause any conflict because no two inputs within set S could produce the whole value as an output of the modulo p product.
- "I.V.," "Initialization vector." A starting value for an iteratively updated cryptographic process.
- "Intractible," "Computationally infeasible." Difficult enough for an attacker to perform, given the computing resources that can reasonably be expected to be

brought to bear in an attack, to provide a desired amount of security or authenticity.

- "Key," "Key value." A number that is processed in combination with one or more input value(s) to produce a cryptographically modified (e.g., encrypted, permuted) output value. Typically, key values are large (e.g., hundreds of binary bits) but small values are shown at various places in this application for ease of illustration. In a pseudogroup, the key value is constrained to be less than p where $p < M$, and to be less than or equal to M where $P > M$.
- "Octave." GNU Software that is largely compatible with MATLAB and freely available on the Internet at the time of this application's filing, e.g., at www.octave.org. (This software was used during software prototype testing.)
- "Optimized efficiency," "Optimized efficiency for a legitimate user." Optimization of software such that intensive numerical operations can be carried out on the computing platform that a legitimate user can be expected to employ for a cryptographic operation with efficiency at least substantially equal to that of any hardware of like complexity. As of this application's filing date, such a computing platform may be a Pentium III-900 computer system, and code optimized for efficiency of legitimate use may include, in a large proportion, 32-bit products using the Pentium's floating-point multiplier and lookups in a 8 KB table. The table can be pre-loaded into the Pentium's on-chip cache, or just as needed, but in any case facilitating fast memory access. An attacker would need to either provide a similar CPU for each branch of a massively parallel hardware attack or emulate the hardware-intensive multiplier and fast memory in a custom chip (e.g., an ASIC).
- "PC," "Computer." A computer available for personal use, i.e., by a person. The term "PC" conventionally refers to an IBM-PC compatible desktop computer. It is used in this application in a broader sense, to include any computer workstation intended for use by a person.

- "Prime." A number that, as determined according to the primality test deemed appropriate for a particular implementation, cannot be divided into factors other than itself and the number one.

- "Pseudogroup." A modulo product operation in discrete mathematics according to various aspects of the inventions that behaves like a multiplicative group, except for relatively infrequent exceptions. The exceptions are the result of the prime modulus of the operation being close to, but not equal to, the set order plus 1. The exceptions are handled, according to particularly advantageous aspects of the invention by (1) if $p > M$, mapping an overflowing output (i.e., greater than the set order) to a hole within the output set in accordance with a deterministically scheme; or (2) if $p < M$, permitting an input value greater than p-1 to pass through the operation unchanged; or (3) other exception handling methods such as permitting the erroneous output to exist and relying on error-correction coding to avoid problems caused by it.

- "Purge," "Purging," "Record destruction." The act of causing a record (paper, electronic, or otherwise) to be destroyed and no longer accessible by anyone. The act becomes irreversible the moment is performed, or shortly thereafter. (When an electronic record is "purged" according to various aspects of the inventions by destruction of a private key, data containing the key may remain in computer memory for a period of time. However, at some point after the purging act has commenced, and before the benefit of purging would be lost if the electronic record were recoverable, the key data becomes overwritten or otherwise destroyed.

- "Salt," "Salt data." Data fixed for a given user and publicly available or derived from publicly available sources that, in combination with secret data, helps prevent pre-compilation of secure delay output values.

- "S-Box." A "substitution box" that permutes an input in a set S1:{1,2,...M1} into an output in a set S2:{1,2,...M2}. (Sets S1 and S2 can be the same.) S-boxes are employed in various embodiments of the present inventions to insert nonlinearities between operations such as the inventive pseudogroup and make the input harder

to predict given the output, or vice versa. An exemplary substitution box for use in a secure delay according to various aspects of the inventions is a lookup table containing encoded digits of pi. All the digits of pi that could be expected to fill up such a lookup table (e.g., within the cache memory limit of a P-III CPU) are known. The digits appear to be randomly distributed. Consequently, such a lookup table provides an effective nonlinearity as well as performing a memory-hogging function to optimized efficiency for legitimate user in a secure delay. A potential attacker would be forced to either re-compute the digits of pi continuously or, somewhat more practically, duplicate the lookup table using RAM that is very expensive in a fast, massively parallel emulation of the secure delay. See Appendix K-2 and the discussion of "optimized efficiency" above.

- "Secure delay." A processing-induced delay interposed between an input and output that transforms a given input value into an unpredictable but deterministic output value. (Both terms are subject to minor deviations from exactness. An output is "unpredictable" when it is computationally infeasible to predict, given computing resources that could reasonably be expected to be brought to bear against the problem. An output is "deterministic" when it is consistently the result given a particular input, except perhaps for rare errors or imperfections in algorithms that do not have a significant negative impact on performance.
- "Sub-Hash." An intermediate hash value, delivered in succession with others, preferably during operation of a secure delay. The sub-hash can be a portion (e.g., MSBs, LSBs, etc.) of a larger (e.g., 160 bit) iteratively computed hash.
- "Tossing an object." In the preferred signature generation embodiment, indicates simply tossing a paper clip onto a piece of paper lying horizontal on a surface so that the resting place of the paper clip is random for all practical purposes. Suitable objects for tossing include, in addition to paper clips, pins, cardboard cutouts, etc. Equivalents to tossing an object include swinging the end of a metal ruler, tapping a pencil point with one's eyes closed, and (where the field of digits is on a computer screen), randomly moving a mouse pointer around.